A METHOD AND SYSTEM OF VOICEXML INTERPRETING

FIELD OF THE INVENTION

[0001] The present invention relates to a VoiceXML Interpreter.

REFERENCE TO COMPUTER PROGRAM LISTING APPENDICES

[0002] Computer program listing appendices are submitted herewith on one compact disc and one duplicate compact disc. The total number of compact discs including duplicates is two. The files on the compact disc are ASCII text files in which the characters are displayed as their corresponding values in hexadecimal format. Their names, dates of creation, directory locations, and sizes in bytes are:

1. Directory appndx-a containing file 42876A.HEX (Appendix A) of June 14, 2001 and of length 44,191 bytes.

2. Directory appndx-b containing file 42876B.HEX (Appendix B) of June 14, 2001 and of length 40,769 bytes.

[0003] The files are referred to herein as appendices A - B respectively. The material on the compact discs is incorporated by reference herein.

BACKGROUND OF INVENTION

[0004] An Interactive Voice Response unit (IVR) is an automated telephone answering system, which responds to a user's telephone call with a voice menu and allows the user to make choices and enter information. IVR systems are widely used in call centers as well as a replacement for human switchboard operators. An IVR system may also integrate database access and fax response.

[0005] IVR systems are typically partnered with an appropriate application to provide services. Traditionally, IVR application developers required an in-depth knowledge of one of numerous proprietary IVR development environments, as well as knowledge of a programming language such as C or C++. The development of these applications was both time consuming and costly. Furthermore, IVRs with speech recognition applications required knowledge of proprietary voice recognition systems

[0006] Recently, the introduction of VoiceXML has alleviated the need of application developers to know the low-level programming necessary to drive the proprietary telephony hardware and speech recognition systems. By enabling the VoiceXML developers to focus on the application and removing the need to understand the low-level programming, the development cycle and the time to market is shortened.

[0007] A VoiceXML environment comprises a VoiceXML document on a VoiceXML document server and a VoiceXML Interpreter. The VoiceXML document specifies each interaction dialog to be conducted by a VoiceXML Interpreter. The user's input can affect the dialogue interpretation.

[0008] Historically, an IVR unit was a closed system. VoiceXML enables IVR applications to take advantage of a distributed IP network approach, enabled through traditional client/server architecture. VoiceXML uses Internet protocols, particularly HTTP in order to access document servers. One problem with client/server interaction is the process involved in accessing the server in order to retrieve documents. This process can often prove to be lengthy and ineffective. One solution VoiceXML offers is to specify multiple interactions per document, thus minimizing client/server interactions.

[0009] An additional solution to the problem of the client/server environment is the use of caching. The VoiceXML Interpreter, just like HTML visual browsers, can use caching to improve performance in fetching documents and other resources. For example, audio recordings, which can be quite large, are as common to VoiceXML documents as images are to HTML pages and therefore should be cached.

[0010] HTML browsers typically perform client side caching, whereby the caching is managed by the client. Server side caching is more difficult; the cache must be maintained for the entire application. Multiple users request documents simultaneously, thereby requiring increased efficiency. One example of efficient server side caching is the PERL® module 'mod_perl' used under the APACHE® web server, which compiles and caches PERL® in scripting languages. PERL® (Practical Extraction and Reporting Language) is a popular script language used in programming. PERL® is often used in an APACHE® environment by linking the PERL® runtime library into the server and providing an object oriented PERL® interface to the server's C language API. These components are bound together by the mod_perl server plug-in, making it possible to write APACHE® modules in PERL®.

[0011] Another aspect of the VoiceXML environment, a function of the fact that it is a scripting language, is its limited extensibility. The VoiceXML Interpreter essentially only runs scripts and functions in accordance with the relevant VoiceXML specification. In order to obtain a level of enhanced functionality, the specification specifies the object tag. An object tag primarily imports outside resources into a web page, such as JAVA® applets, multimedia files, web plug-ins or ACTIVEX® components. Object tags enable the addition of special functions and local functions external to the current VoiceXML specification. An example of one such implementation of object tags is NUANCE's®, which employs a virtual JAVA® machine. Through the use of tags, the application developer can specify JAVA® speech objects (modules), which are loaded into the JAVA® virtual machine and provide enhanced functionality.

[0012] The VoiceXML environment adds complexity to IVR systems. This complexity makes it harder to fulfill the requirement for high availability, whereby the IVR user interacting with the system should experience as little interference and malfunction as possible.

SUMMARY OF THE INVENTION

[0013] This invention seeks to provide a system and methodology for a flexible, robust and scalable VoiceXML Interpreter, which provides efficient rendering of VoiceXML documents.

[0014] There is thus provided in accordance with a preferred embodiment of the present invention a VoiceXML interpreting system including a VoiceXML Interpreter. The VoiceXML interpreter includes a Fetcher operative to retrieve documents, a compiler operative to compile documents retrieved by the Fetcher and a cache which stores compiled documents compiled by the compiler.

[0015] There is further provided in accordance with a preferred embodiment of the present invention a VoiceXML interpreting method. The VoiceXML interpreting method includes retrieving documents, compiling retrieved documents and caching compiled documents compiled by the compiler.

[0016] Further in accordance with a preferred embodiment of the present invention the VoiceXML interpreting system also includes Storage Device, which stores

state information, related to execution of the compiled documents. Preferably, the VoiceXML interpreting system further includes a backup VoiceXML Interpreter communicating with the Storage Device. The Storage Device preferably includes a memory database external to the VoiceXML Interpreter and to the backup Interpreter.

[0017] There is also provided in accordance with a preferred embodiment of the present invention a VoiceXML interpreting system. The VoiceXML interpreting system includes a Fetcher operative to retrieve documents and a Storage Device, which stores state information related to execution of the documents.

[0018] Further in accordance with a preferred embodiment of the present invention the VoiceXML interpreting system also includes a backup VoiceXML Interpreter communicating with the Storage Device. Preferably, the Storage Device includes a memory database external to the VoiceXML Interpreter and to the backup Interpreter.

[0019] There is further provided in accordance with another preferred embodiment of the present invention a VoiceXML interpreting system. The VoiceXML interpreting system includes a communication device, telephony hardware communicating with the communication device, a switchboard communicating with the telephony hardware, a VoiceXML Interpreter communicating with the switchboard and an object module communicating with the at least one of the telephony hardware, switchboard and VoiceXML Interpreter.

[0020] There is also provided in accordance with yet another preferred embodiment of the present invention a VoiceXML interpreting method. The VoiceXML interpreting method includes retrieving documents and storing state information related to execution of the documents.

[0021] Further in accordance with a preferred embodiment of the present invention the object module includes a dynamically loadable library. Preferably, the dynamically loadable library is operative to allocate telephone resources.

[0022] There also is provided in accordance with yet a further preferred embodiment of the present invention a VoiceXML interpreting system. The VoiceXML interpreting system includes a communication device, telephony hardware communicating with the communication device, a switchboard communicating with the telephony hardware, a VoiceXML Interpreter communicating with the switchboard and

an object module communicating with the at least one of the telephony hardware, switchboard and VoiceXML Interpreter.

[0023] Further in accordance with a preferred embodiment of the present invention the object module includes a dynamically loadable library. Preferably, the dynamically loadable library is operative to allocate telephone resources.

BRIEF DESCRIPTION OF THE DRAWINGS AND APPENDICES

[0024] The present invention will be more fully understood and appreciated from the following detailed description, taken in conjunction with the following drawings and appendices in which:

Fig 1 is a simplified block diagram illustration of a prior art VoiceXML environment, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig 2. is a simplified block diagram illustrating the structure and operation of the VoiceXML environment of Fig 1, in accordance with a preferred embodiment of the present invention;

Fig 3. is a simplified flow chart, illustrating operation of the system of figs. 1 and 2 in accordance with a preferred embodiment of the present invention;

Fig. 4 is a simplified block diagram illustrating the architecture involved in extracting state information from the VoiceXML Interpreter; and

Fig.5 is a simplified block diagram illustrating the implementation of object tag extensions, in accordance with a preferred embodiment of the present invention.

DESCRIPTION OF COMPUTER LISTING APPENDICES

[0025] Appendix A is a sample of a DLL file used to load a random number generator application, in accordance with a preferred embodiment of the present invention; and

Appendix B is a sample of a DLL file used to load a DTMF generator application, in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

[0026] The present invention provides a system and methodology for a flexible, robust and scalable VoiceXML Interpreter, which provides efficient rendering of VoiceXML documents. Prior art modules such as mod_perl cannot remotely fetch documents and are not able to work in either XML environments or telephony environments. Whereas prior art VoiceXML systems can cache fetched resources, the present invention improves efficiency by caching compiled and optimized internal representations of raw fetched data. Furthermore, in accordance with the present invention, the caching of complied data is performed in an environment capable of simultaneous handling of multiple client requests through the same server.

[0027] In accordance with the present invention, the raw fetched data is first preprocessed into a ready-to-use binary format, rather than raw data directly available from the fetching transaction. When the data is in this binary format, the data can be used concurrently by multiple users of the application. This saves on processing time, as there is only one conversion per cached resource and only one binary copy shared by many users within the application. This also saves on memory because the optimized data may reduce the volume of the data.

[0028] Reference is now made to Fig. 1, which is a simplified block diagram illustration of a prior art VoiceXML environment. In the example shown, a user 100 wishing to access web-based information such as general news items using the telephone, typically places a telephone call to an IVR 102. The IVR 102, containing a VoiceXML Interpreter 104, queries a VoiceXML document server 106 for information and the VoiceXML document server 106 accesses the information from a Storage Device 108. The VoiceXML Interpreter 104 may request a welcome page 109. The VoiceXML Interpreter 104 may retrieve the welcome page 109 from the Storage Device 108 within the VoiceXML document server 106 and may run the script embedded in the welcome page 109. The script may reference a recorded welcome message in the form of a Wav file. Alternatively the script may be a simple piece of 'welcome' text that is sent to an external Text-to-Speech server 110 and converted into audio by the VoiceXML Interpreter 104. The welcome message may then be played back to the user 100 and the VoiceXML Interpreter 104 may await user input.

[0029] Next, the user 100 may communicate a speech command via the telephone such as "May I please have the latest stock quotes". The VoiceXML Interpreter 104 receives the speech and relays the speech to an Automated Speech Recognition server (ASR) 112. The Automated Speech Recognition server 112 recognizes the user's speech and identifies what the user said. The Automated Speech Recognition server 112 may have a limited number of recognized responses and, depending on the user's input, may send back an appropriate response to the VoiceXML Interpreter 104. In this fashion, a dialogue exists between the VoiceXML Interpreter 104 and the user 100. Alternatively, a DTMF input is used instead of speech.

[0030] Reference is now made to Fig. 2, which is a simplified block diagram illustrating the structure and operation of an improved VoiceXML Interpreter 204, constructed and operative in accordance with a preferred embodiment of the present invention and having optimized caching. Reference is also made to Fig. 3, which is a simplified flowchart illustrating operation of the system of Fig. 2 in an environment of Fig. 1, in accordance with a preferred embodiment of the present invention.

[0031] As seen in Figs. 2 and 3, a user 200 places a telephone call to an IVR 202. The IVR 202 connects to the VoiceXML Interpreter 204. The VoiceXML Interpreter 204 activates an Execution Thread 206 to begin processing the call. The Execution Thread 206 sends a request for an initial document to a Fetcher 208. The Fetcher 208 requests a copy of the document from a Cache 210. If the Cache 210 already contains a copy of the requested document, the Fetcher 208 retrieves a copy of the document from the Cache 210 and sends it to the Execution Thread 206, which executes the script.

[0032] If the Cache 210 does not contain a copy of the requested document, the Fetcher 208 connects to a VoiceXML Document server 212 to retrieve a raw copy of the document. The Fetcher 208 sends the raw document to a Compiler 214, which comprises a Lexographical Analyzer 216, a Parser 218, a Code Generator 220 and an Optimizer 222. The Lexographical Analyzer 216 analyzes the text and generates 'tokens'. These tokens are sent to the Parser 218, where they are syntactically analyzed. The Parser 218 then sends the tokens to the Code Generator 220, where the tokens are mapped to a binary code that is executable. The executable code could be of various types, such as ASCII, EBCDIC or JAVA® byte codes.

[0033] Once the code is deemed executable, it is relayed to the Optimizer 222. The Optimizer 222 optimizes the code and prepares the code for use. The prepared code is sent to the Fetcher 208. The Fetcher 208 places a copy of the prepared code in the Cache 210 and sends a reference of the prepared code to the Execution Thread 206.

[0034] In a preferred embodiment of the present invention, the prepared executable code is saved in memory. In another preferred embodiment of the present invention, the prepared executable code is stored on an external device, such as a disk.

[0035] Reference is now made to Fig. 4, which is a simplified block diagram illustrating a preferred architecture involved in extracting state information from the VoiceXML Interpreter 204. The current state of the VoiceXML Interpreter 204 is abstracted to enable high availability of data. In order to protect data in the event of a crash, the current state information is offloaded onto a Memory Database 400 that can be accessed by a Back-up Interpreter 402, which may take over from the VoiceXML Interpreter 204 in the event that the primary IVR 206 fails.

[0036] Reference is now made to Fig. 5, which is a simplified block diagram illustrating one implementation of object tag extensions. In one embodiment of the present invention, an object tag is used to enable fax transmission. For example, a user 500 who wishes to send a fax, makes a call to the IVR 502. Telephony Hardware 504, located within the IVR 502 detects the telephone call and notifies a Switchboard 506. The Switchboard 506 instructs the Telephony Hardware 504 to answer the call. After the Telephony Hardware 504 answers the call, the Switchboard 506 queries the VoiceXML Interpreter 508 for further instructions. The VoiceXML Interpreter 508 may fetch the application from the Document Server 510, which queries the user 500 for the destination fax address and the content of the fax. For example, the application may play a Wav file though the VoiceXML Interpreter 508.

[0037] Having received a destination fax address/number and fax content from the user 500, the VoiceXML Interpreter 508 may execute the object tag that instructs the transmission of the fax. In one embodiment of the present invention, the Object Module 512 refers to a Dynamic Link Library (DLL). The VoiceXML Interpreter 508 loads the DLL, which is typically found and loaded to memory. In another embodiment of the present invention the Object Module 512 is a UNIX® shared object library. In

another embodiment of the present invention, the library may be a dynamically loadable library or a static library.

[0038] The VoiceXML Interpreter 508 invokes an 'execute' function in the Object Module 512 and communicates parameters, such as the fax number and an on-disk reference to the content of the fax. The VoiceXML Interpreter 508 also communicates the access point to the Telephony Hardware 504, which enables the loaded library to directly access the Telephony Hardware 504. For instance, direct access to the Telephony Hardware 504 could enable the library to directly allocate telephone resources, such as a telephone line, for fax transmission.

[0039] Additionally, the Object Module 512 may send back a response, such as an error result, to the user 500 via Text-to-Speech Server (TTS) 514. Furthermore, the Object Module 512 may utilize the ASR 516 functionality. For example, the ASR 516 may convert a recorded message to text, for fax transmission.

[0040] It will be appreciated by persons skilled in the art that the present invention is not limited by what has been particularly shown and described hereinabove. Rather the present invention includes combinations and sub-combinations of the various features described hereinabove as well as modifications and extensions thereof, which would occur to a person skilled in the art and which do not fall within the prior art.